SISTEMI OPERATIVI

ESERCIZIO N. 1 del 25 GIUGNO 2001

In una **catena di produzione** vi sono **N processi assemblatori** che necessitano di risorse riutilizzabili di tipo A e B e di risorse consumabili di tipo C, in quantità variabile a seconda del processo. Le risorse riutilizzabili di tipo A e B sono presenti in quantità limitata, rispettivamente MAXA e MAXB, mentre le risorse di tipo C sono prodotte da un **processo produttore** che le deposita in un magazzino che può contenere al massimo MAXC risorse. Ad esempio, un processo assemblatore potrebbe aver bisogno di 3 A, 4 B e 8 C, mentre un altro di 10 A, 3 B e 6 C; al termine il primo restituirà 3 A e 4 B, mentre il secondo 10 A e 3 B.
Si implementi una soluzione usando il costrutto monitor per modellare la **catena di produzione** e i processi per modellare i **processi assemblatori** e il **processo produttore** e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program fabbrica

const     MAXA = ...; { numero di risorse A }
const     MAXB = ...; { numero di risorse B }
const     MAXC = ...; { capacità del magazzino }

type proc_ass = process
          (qa: 1..MAXA, qb: 1..MAXB, qc: 1..MAXC)
begin
     repeat
          catena.richiedi (qa, qb, qc);
          <assembla il pezzo>
          catena.rilascia(qa, qb);
     until false
end

type proc_pro = process
begin
     repeat
          <produci risorsa C >
          catena.deposita;
     until false
end

type catena_produzione = monitor

{ variabili del monitor }
var  risA: 0..MAXA;
     { numero di risorse A disponibili }
     risB: 0..MAXB;
     { numero di risorse B disponibili }
     risC: 0..MAXC;
     { numero di risorse C disponibili }
     incoda: integer;
     { numero di processi assemblatori sospesi }
     coda : condition;
```

```
        { coda su cui sospendere i processi assemblatori }
        codaProd : condition;
        { coda su cui sospendere il processo produttore }

procedure entry richiedi (qa: integer, qb: integer, qc:
integer)
begin
        { se non ci sono tutte le risorse richieste }
        while (risA<qa or risB<qb or risC<qc) do
          { mi sospendo }
          begin incoda++;  coda.wait; incoda--; end;
        { acquisisco le risorse }
        risA := risA - qa;
        risB := risB - qb;
        risC := risC - qc;
        { ho diminuito il numero di risorse C in magazzino }
        { quindi risveglio il processo produttore se è sospeso }
        if codaProd.queue
        then codaProd.signal;
end

procedure entry rilascia (qa: integer, qb: integer)
var s, i: integer;
begin
        { rilascio le risorse }
        risA := risA + qa;
        risB := risB + qb;
        { sveglio tutti i processi assemblatori in coda }
        s := incoda;
        for i := 1 to s do
                coda.signal;
end
```

```
procedure entry deposita
var s, i: integer;
begin
      { se non c'è posto in magazzino }
      if risC = MAXC
      then codaProd.wait; { mi sospendo }
      { deposito una risorsa di tipo C }
      risC++;
      { sveglio tutti i processi assemblatori in coda }
      s := incoda;
      for i := 1 to s do
            coda.signal;
end

begin { inizializzazione delle variabili }
      risA = MAXA;
      risB = MAXB;
      risC = 0; { all'inizio il magazzino C è vuoto }
end

var  catena: catena_produzione; { il nostro monitor }
      p1, p2, ... : proc_ass (k, l, m);
      p: proc_pro;

begin end.
```

Note

Per semplicità, si è supposto che il processo produttore produca 1 risorsa C alla volta.

Starvation

I processi che necessitano di molte risorse rischiano di essere sempre superati da quelli che ne necessitano meno.